AMENDMENTS TO THE CLAIMS:

This listing of claims will replace all prior versions, and listings, of claims in the application:

LISTING OF CLAIMS:

1. (Currently Amended) A method for modifying serial dependencies in a procedure, said method comprising:

building a graph representation of said procedure, said graph representation having an origin and including a unique position, relative to said origin, for each memory operation in said procedure;

designating a location type for each memory operation in said graph representation; each said location type based on a characteristic of said corresponding memory operation;

generating a summary for each memory operation in the graph representation that indicates for each location type used in the procedure the closest preceding memory operation to affect that location type;

identifying a first memory operation having the same location type as a second memory operation; wherein said first memory operation is positioned closer to said origin than said second memory operation, and said graph representation does not include any additional memory operations of the same location type between the first and second memory operations; and

moving said second memory operation to a new position in said graph representation that is closer to said first memory operation, wherein the moving step

includes (i) removing one or more of the serial dependencies in said initial set of serial dependencies that is associated with said second memory operation and (ii) creating a new serial dependency between said first memory operation and said second memory operation.

2. (Currently Amended) The method of claim 1 wherein

the building step includes the step of assigning an initial set of serial dependencies between program operations represented in said graph representation~~; and~~

~~the moving step includes (i) removing one or more of the serial dependencies in said initial set of serial dependencies that is associated with said second memory operation and (ii) creating a new serial dependency between said first memory operation and said second memory operation~~.

3. (Original) The method of claim 1 wherein said graph representation is an intermediate representation.

4. (Original) The method of claim 3 wherein said intermediate representation is a static single assignment graph embedded in a control flow graph.

5. (Original) The method of claim 1 wherein said moving step results in said second memory operation being advanced in a schedule of machine instructions.

6. (Original) The method of claim 1 wherein said first memory operation is positioned before a repetitive loop in said procedure and said second memory operation is within said repetitive loop; said graphic representation including a phi node that corresponds to a loop back position in said repetitive loop;

said designating step further comprising a step of advancing through said repetitive loop in order to determine a location type for each memory operation in said repetitive loop;

wherein

when a memory operation having the same location type as said first and second memory operation exists in said loop, said new position in said graph representation is a position that is serially dependent upon said phi node; and

when a memory operation having the same location type as said first and second memory operation does not exist in said loop, said new position in said graph representation is a position that is not serially dependent on any operation in the loop.

7. (Original) The method of claim 1 wherein said first memory operation is a store or an array store and said second memory operation is a load or an array load.

8. (Original) The method of claim 1 wherein said procedure includes a calling procedure and said first memory operation is in said calling procedure and said second memory operation is in a called procedure that is called by an operation in said calling procedure.

9. (Original) The method of claim 8 wherein said moving step results in a placement of said second memory operation in said calling procedure.

10. (Original) The method of claim 1 wherein said location type of each memory operation is selected from the group consisting of a predefined set of base types, a predefined set of array types, object types, and object field types.

11. (Original) The method of claim 1 wherein the building step further comprises adding a global store dependency to each operation in said procedure that reads a variable from or stores a variable to memory; the method further comprising:

generating a schedule of machine instructions in accordance with said graph representation, wherein each said machine instruction in said schedule of machine instructions, which corresponds to an operation that reads a variable from or stores a variable to memory, is ordered in accordance with said global store dependency associated with said operation.

12. (Original) The method of claim 1 wherein a first operation affects a value of a variable stored in memory and a second operation serially follows said first operation, said building step further comprising adding a global store dependency from said second operation to said first operation; the method further comprising:

generating a schedule of machine instructions in accordance with said graph representation, wherein said machine instructions in said schedule of machine

instructions corresponding to said second operation are scheduled after said machine instructions corresponding to said first operation.

13. (Currently Amended) A computer program product for use in conjunction with a computer system, the computer program product capable of modifying serial dependencies in a procedure, the computer program product comprising a computer readable storage medium and a computer program mechanism embedded therein, comprising:

instructions for building a graph representation of said procedure, said graph representation having an origin and including a unique position, relative to said origin, for each memory operation in said procedure;

instructions for designating a location type for each memory operation in said graph representation; each said location type based on a characteristic of said corresponding memory operation;

instructions for generating a summary for each memory operation in the graph representation that indicates for each location type used in the procedure the closest preceding memory operation to affect that location type;

instructions for identifying a first memory operation having the same location type as a second memory operation; wherein said first memory operation is positioned closer to said origin than said second memory operation, and said graph representation does not include any additional memory operations of the same location type between said first and second memory operations; and

instructions for moving the second memory operation to a new position in said graph representation that is closer to said first memory operation, wherein the

instructions for moving include (i) instructions for removing one or more of the serial dependencies in said initial set of serial dependencies that is associated with said second memory operation and (ii) creating a new serial dependency between said first memory operation and said second memory operation.

14. (Currently Amended) The computer program product of claim 13 wherein:

the instructions for building include instructions for assigning an initial set of serial dependencies between program operations represented in the graph~~; and~~

~~the instructions for moving include (i) instructions for removing one or more of the serial dependencies in said initial set of serial dependencies that is associated with said second memory operation and (ii) creating a new serial dependency between said first memory operation and said second memory operation~~.

15. (Original) The computer program product of claim 13 wherein said graph representation is an intermediate representation.

16. (Original) The computer program product of claim 15 wherein said intermediate representation is a static single assignment graph embedded in a control flow graph.

17. (Original) The computer program product of claim 13 wherein said moving step results in said second memory operation being advanced in a schedule of machine instructions.

18. (Original) The computer program product of claim 13 wherein said first memory operation is positioned before a repetitive loop in said procedure and said second memory operation is within said repetitive loop; said graphic representation including a phi node that corresponds to a loop back position in said repetitive loop;

said instructions for designating further comprising instructions for advancing through said repetitive loop in order to determine a location type for each memory operation in said repetitive loop; wherein

when a memory operation having the same location type as said first and second memory operation exists in said loop, said new position in said graph representation is a position that is serially dependent upon said phi node; and

when a memory operation having the same location type as said first and second memory operation does not exist in said loop, said new position in said graph representation is a position that is not serially dependent on any operation in the loop.

19. (Original) The computer program product of claim 13 wherein said first memory operation is a store or an array store and said second memory operation is a load or an array load.

20. (Original) The computer program product of claim 13 wherein said procedure includes a calling procedure and said first memory operation is in said calling procedure and said second memory operation is in a called procedure that is called by an operation in said calling procedure.

21. (Original) The computer program product of claim 20 wherein said moving step results in a placement of said second memory operation in said calling procedure.

22. (Original) The computer program product of claim 13 wherein said location type of each memory operation is selected from the group consisting of a predefined set of base types, a predefined set of array types, object types, and object field types.

23. (Original) The computer program product of claim 13 wherein the building step further comprises adding a global store dependency to each operation in said procedure that reads a variable from or stores a variable to memory; the computer program mechanism further comprising:

instructions for generating a schedule of machine instructions in accordance with said graph representation, wherein each said machine instruction in said schedule of machine instructions, which corresponds to an operation that reads a variable from or stores a variable to memory, is ordered in accordance with said global store dependency associated with said operation.

24. (Original) The computer program product of claim 13 wherein a first operation affects a value of a variable stored in memory and a second operation serially follows said first operation, said building step further comprising adding a

global store dependency from said second operation to said first operation; the computer program mechanism further comprising:

instructions for generating a schedule of machine instructions in accordance with said graph representation, wherein said machine instructions in said schedule of machine instructions corresponding to said second operation are scheduled after said machine instructions corresponding to said first operation.

25. (Currently Amended) A computer system for modifying serial dependencies in a procedure, comprising:

a memory to store instructions and data;

a processor to execute the instructions stored in the memory;

the memory storing:

instructions for building a graph representation of said procedure, said graph representation having an origin and including a unique position, relative to said origin, for each memory operation in said procedure;

instructions for designating a location type for each memory operation in said representation; each said location type based on a characteristic of said corresponding memory operation;

instructions for generating a summary for each memory operation in the graph representation that indicates for each location type used in the procedure the closest preceding memory operation to affect that location type;

instructions for identifying a first memory operation having the same location type as a second memory operation; wherein said first memory operation is positioned closer to said origin than said second memory operation, and said graph

representation does not include any additional memory operations of the same location type between the first and second memory operations; and

instructions for moving said second memory operation to a new position in said graph representation that is closer to said first memory operation, wherein the instructions for moving include instructions for removing one or more of the serial dependencies in said initial set of serial dependencies and creating a new serial dependency from the first memory operation to the second memory operation.

26. (Currently Amended) The computer system of claim 25 wherein:

the instructions for building include instructions for assigning an initial set of serial dependencies between program operations represented in the graph~~; and~~

~~the instructions for moving include instructions for removing one or more of the serial dependencies in said initial set of serial dependencies and creating a new serial dependency from the first memory operation to the second memory operation~~.

27. (Original) The computer system of claim 25 wherein said graph representation is an intermediate representation.

28. (Original) The computer system of claim 27 wherein said intermediate representation is a static single assignment graph embedded in a control flow graph.

29. (Original) The computer system of claim 25 wherein said instructions for moving results in said second memory operation being advanced in a schedule of machine instructions.

30. (Original) The computer system of claim 25 wherein said first memory operation is positioned before a repetitive loop in said procedure and said second memory operation is within said repetitive loop; said graphic representation including a phi node that corresponds to a loop back position in said repetitive loop;

said instructions for designating further comprising instructions for advancing through said repetitive loop in order to determine a location type for each memory operation in said repetitive loop; wherein

when a memory operation having the same location type as said first and second memory operation exists in said loop, said new position in said graph representation is a position that is serially dependent upon said phi node; and

when a memory operation having the same location type as said first and second memory operation does not exist in said loop, said new position in said graph representation is a position that is not serially dependent on any operation in the loop.

31. (Original) The computer system of claim 25 wherein said first memory operation is a store or an array store and said second memory operation is a load or an array load.

32. (Original) The computer system of claim 25 wherein said procedure includes a calling procedure and said first memory operation is in said calling procedure and said second memory operation is in a called procedure that is called by an operation in said calling procedure.

33. (Original) The computer system of claim 25 wherein said instructions for moving results in a placement of said second memory operation in said calling procedure.

34. (Original) The computer system of claim 25 wherein said location type of each memory operation is selected from the group consisting of a predefined set of base types, a predefined set of array types, object types, and object field types.

35. (Original) The computer system of claim 25 wherein the instructions for building further comprises adding a global store dependency to each operation in said procedure that reads a variable from or stores a variable to memory; the memory further storing:

instructions for generating a schedule of machine instructions in accordance with said graph representation, wherein each said machine instruction in said schedule of machine instructions, which corresponds to an operation that reads a variable from or stores a variable to memory, is ordered in accordance with said global store dependency associated with said operation.

36. (Original) The computer system of claim 25 wherein a first operation affects a value of a variable stored in memory and a second operation serially follows said first operation, said building step further comprising adding a global store dependency from said second operation to said first operation; the memory further storing:

instructions for generating a schedule of machine instructions in accordance with said graph representation, wherein said machine instructions in said schedule of machine instructions corresponding to said second operation are scheduled after said machine instructions corresponding to said first operation.

37. (Previously Presented) The method of claim 1 wherein when no known preceding memory operation in the procedure affects a location type, said summary indicates the origin for that location type.

38. (Previously Presented) The computer program product of claim 13 wherein when no known preceding memory operation in the procedure affects a location type, said summary indicates the origin for that location type.

39. (Previously Presented) The computer system of claim 25 wherein when no known preceding memory operation in the procedure affects a location type, said summary indicates the origin for that location type.